Surface Splatting

Matthias Zwicker
Markus Gross

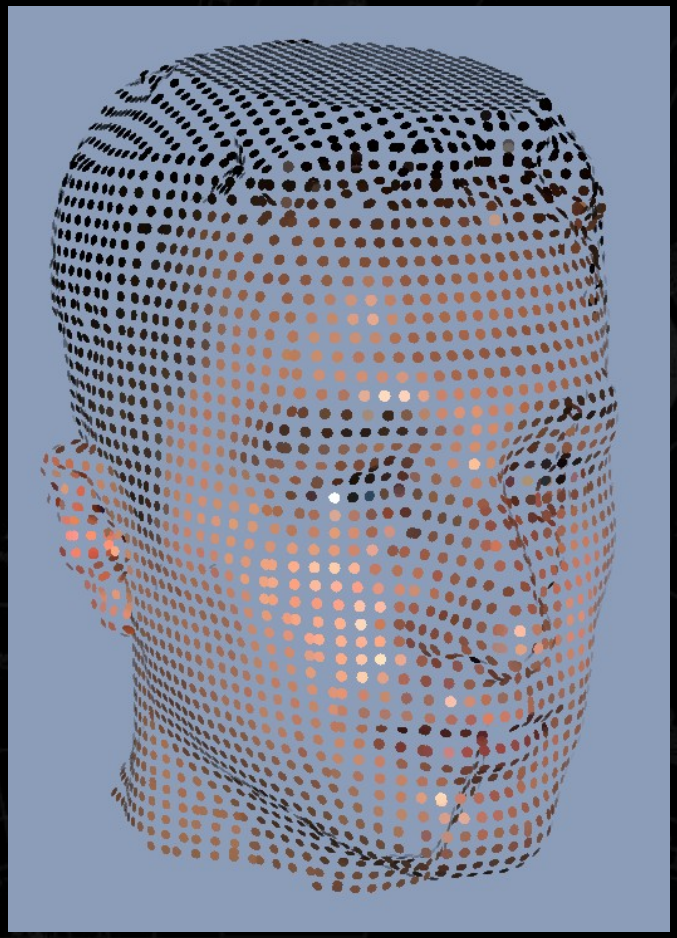
Hanspeter Pfister
Jeroen van Baar

ETH Zurich

Outline

- Introduction
- Related work
- Technical discussion
- Further issues
- Conclusion

Point Rendering



- Surfaces are represented as a set of points without connectivity information
- Points store several surface attributes (*surfels*)
- To render, forward project each point separately

Point vs. Polygon Rendering

Points

- Efficient for highly complex models
- Fast preprocessing
- Ad hoc texture filtering and image reconstruction

Polygons

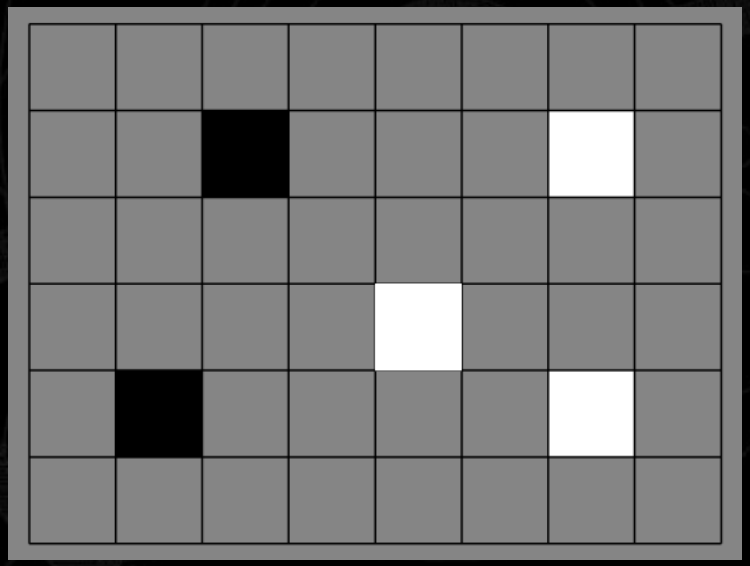
- Good for flat or slightly curved surfaces
- Costly mesh generation and LOD data structures
- High-quality texture filtering algorithms available

Image Reconstruction

- Generate a raster image from projected points
- Similar to polygon rasterization: *Sample projected rendering primitives at output pixel locations*
- Avoid sampling artifacts (holes, aliasing)

Sampling Artifacts

screen space



pixel sampling

Pixel Sampling

minification

magnification

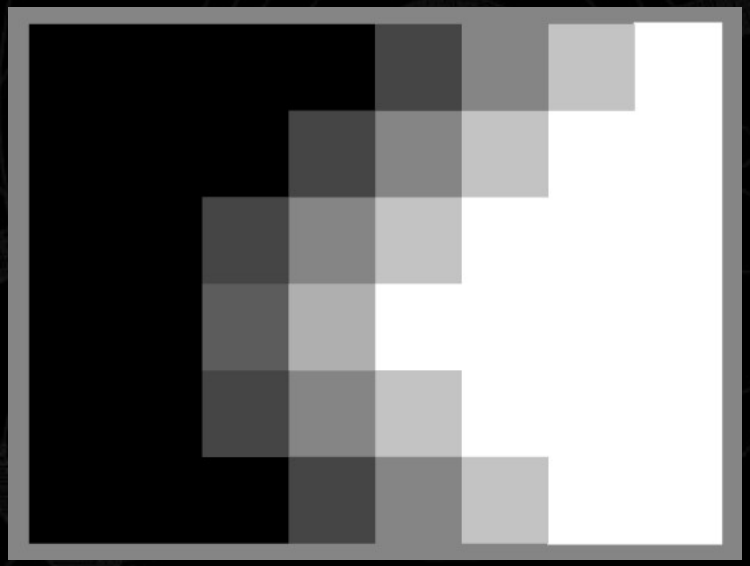
128 x 192

aliasing

holes

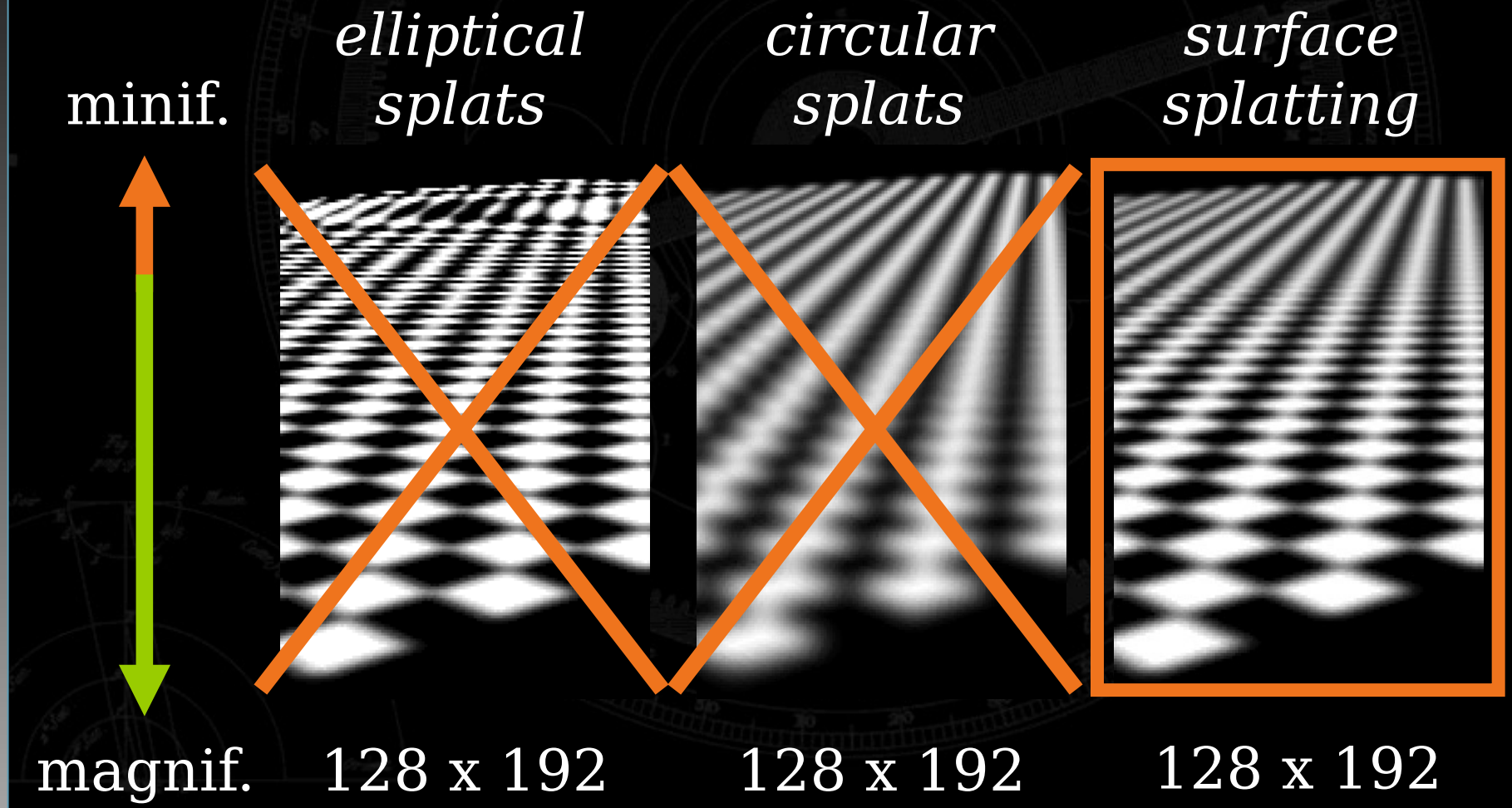
Splatting

screen space



splatting

Splatting Comparison



Related Work

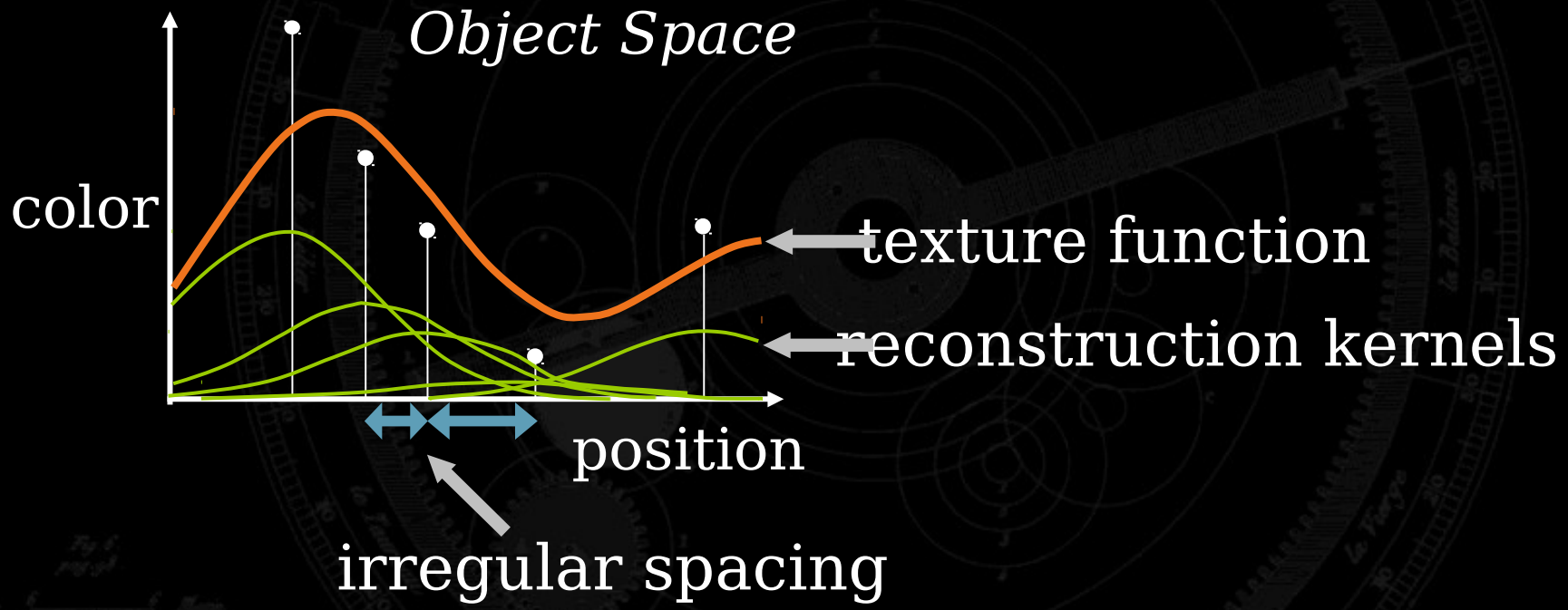
Point Rendering
- Levoy et al. 1986
- Rusinkiewicz et al. 2000
- Pfister et al. 2000

Volume Rendering

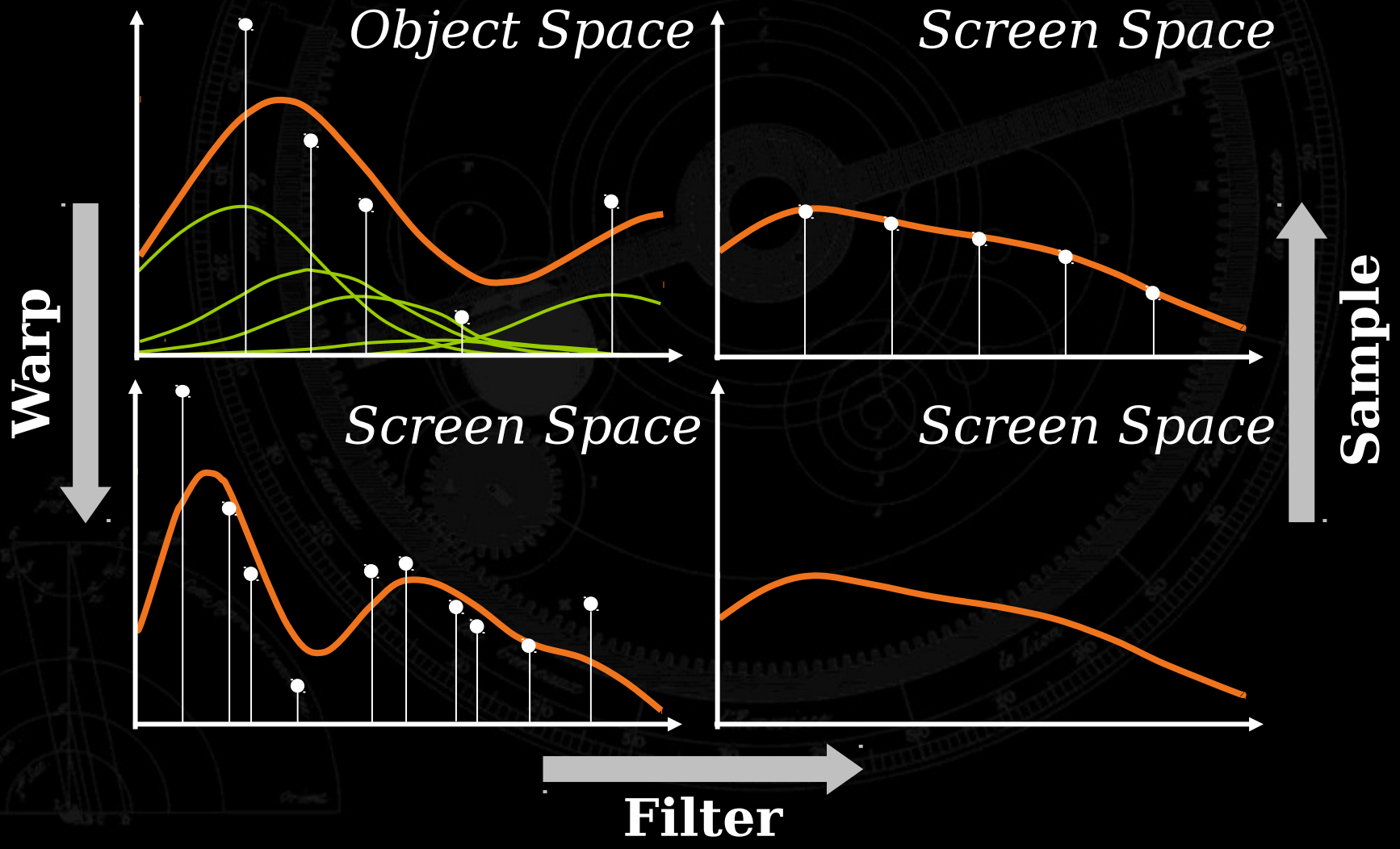
Image-based Rendering

Texture Mapping
- Greene, Heckbert 1986
- Heckbert 1989
  EWA texture filter

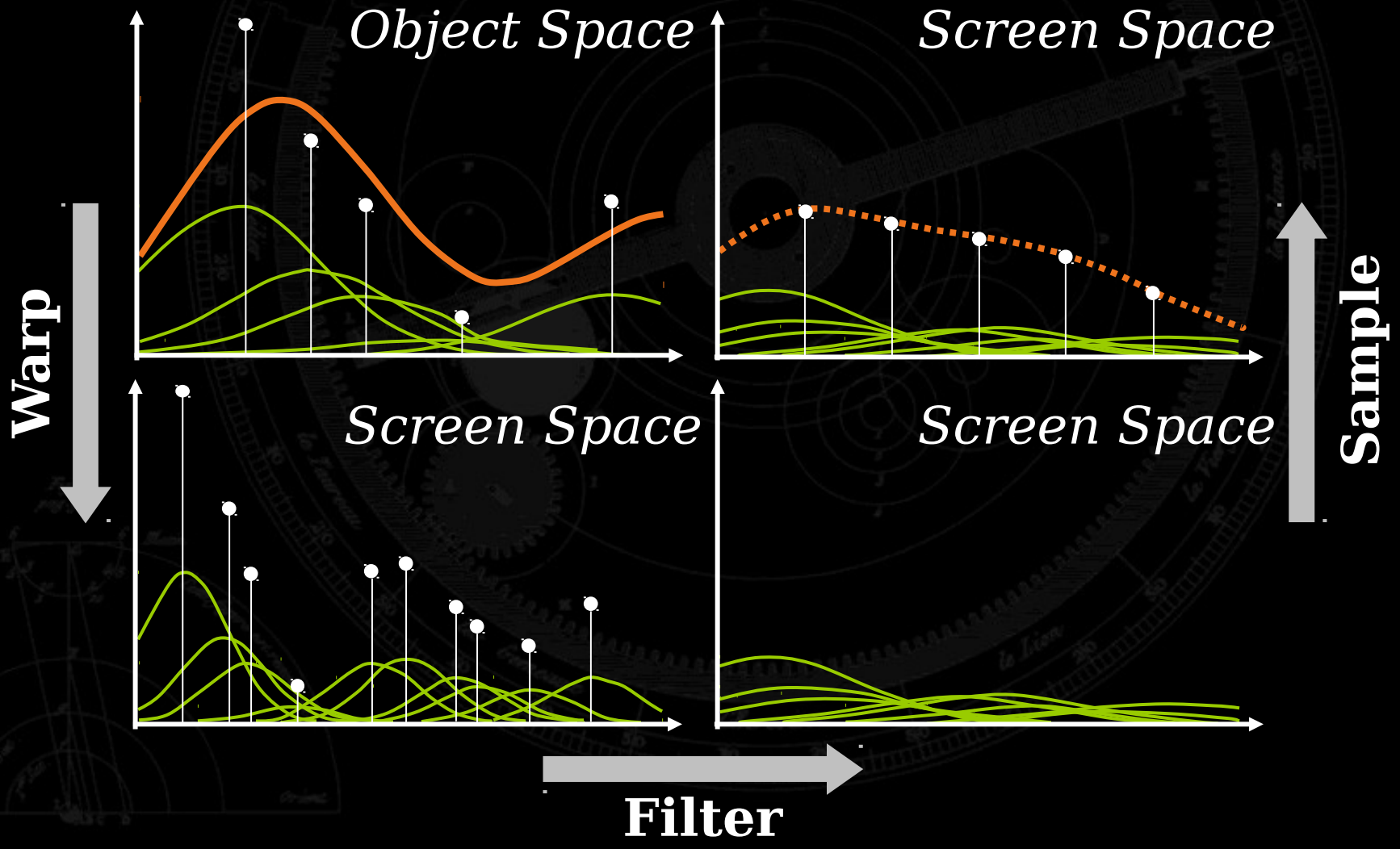
The Surface Splatting Framework: 1D



The Surface Splatting Framework: 1D



The Surface Splatting Framework: 1D

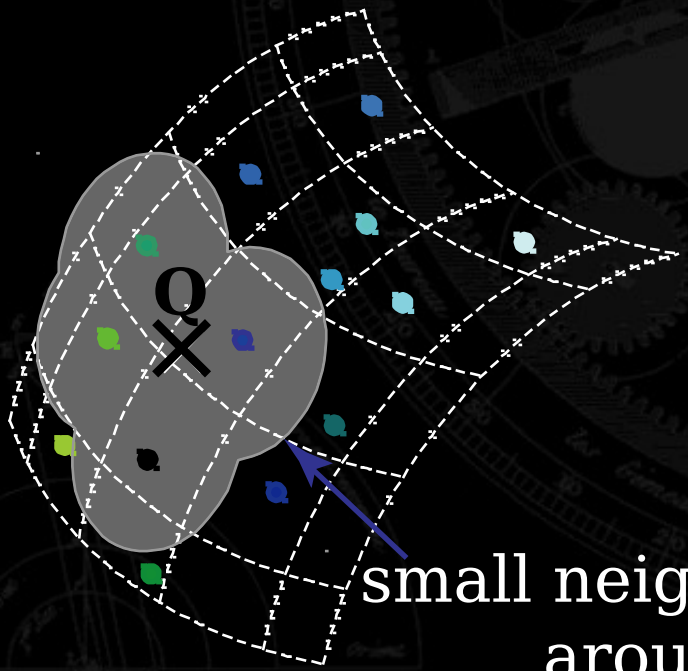


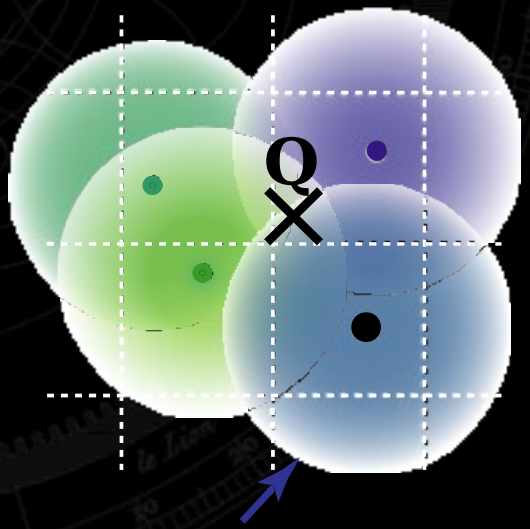
2D Texture Function

local parameterization

3D object space

2D parameterization

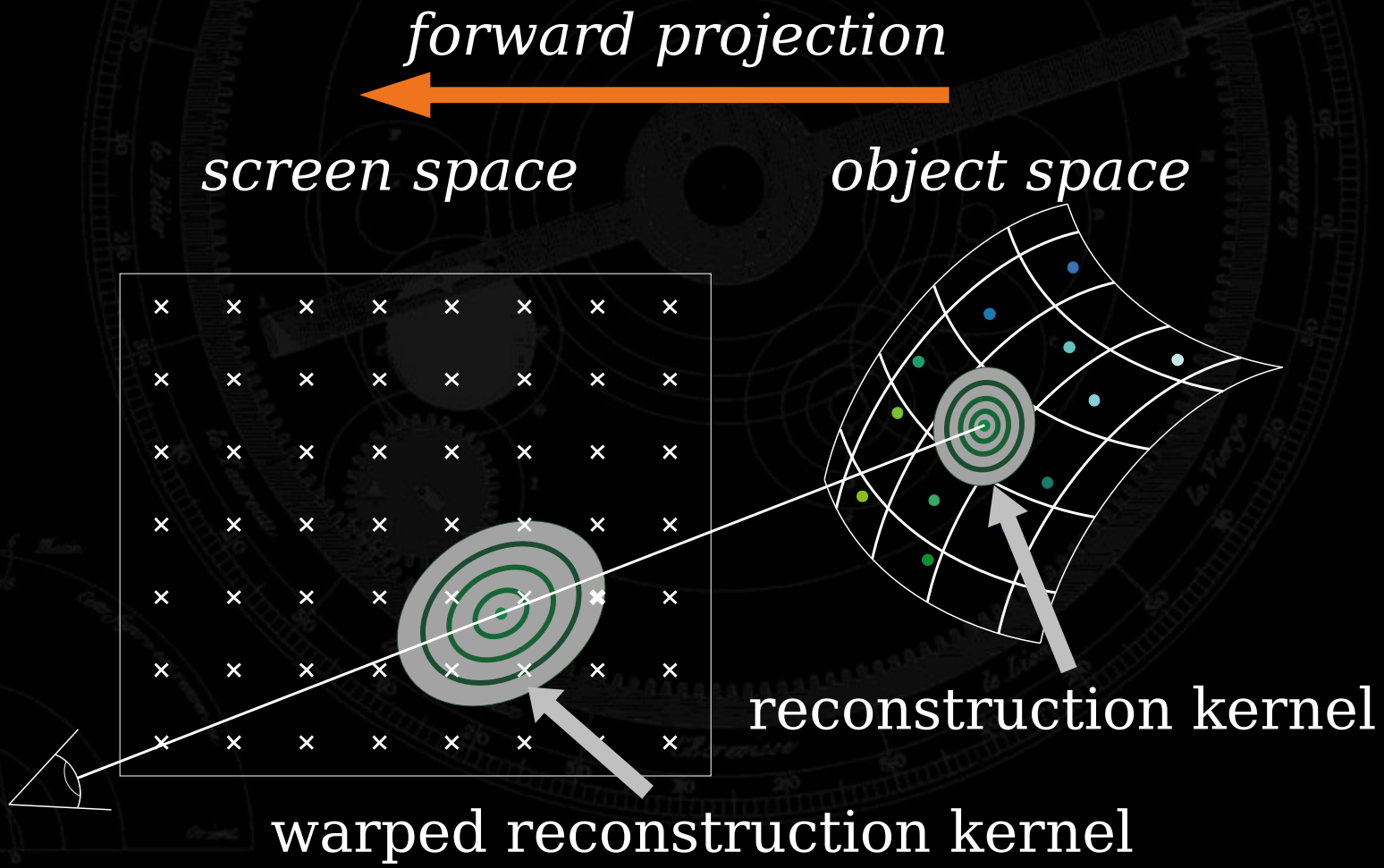




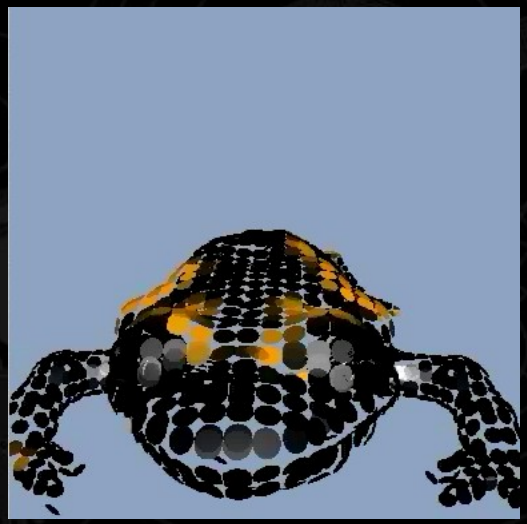
small neighborhood around **Q**

reconstruction kernel

Warping the 2D Texture Function



Projecting the Reconstruction Kernels



Mathematical Formulation

$$g(x) = \sum_k w_k r_k(m^{-1}(x)) \otimes h(x)$$

screen space resampling filter

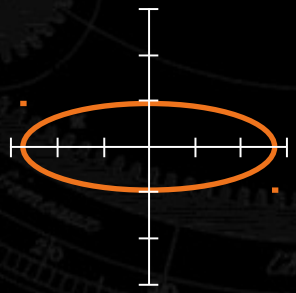
- The *screen space* resampling filter combines a *warped reconstruction kernel* and a *low-pass filter*
- The *screen space* formulation is inverse to Heckbert's *source space* resampling filter

Gaussian Kernels

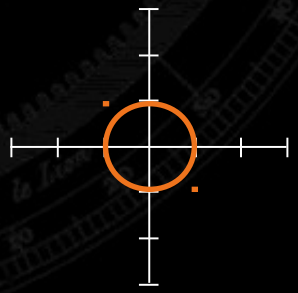
$$g(x) = \sum_k w_k r_k(m^{-1}(x)) \otimes h(x)$$

Gaussian reconstruction kernel

Gaussian low-pass filter



screen space



screen space

Gaussian Kernels

- Closed under affine mappings and convolution

$$g(x) = \sum_k w_k r_k(m^{-1}(x)) \otimes h(x)$$

$$= \sum_k w_k G_k(x)$$

Gaussian resampling filter

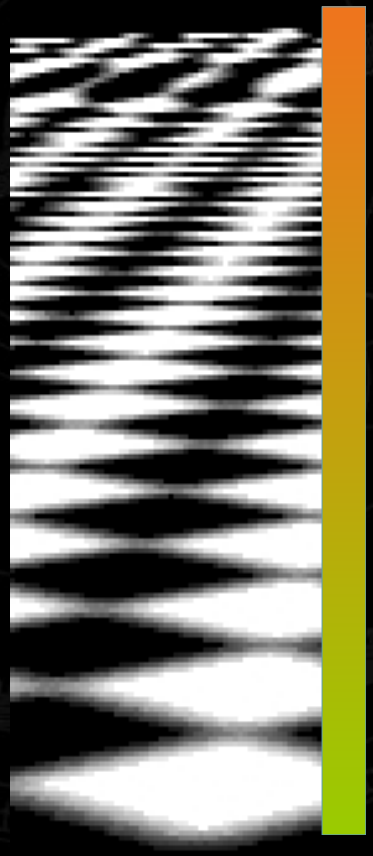
"screen space EWA"

- Analytic expression of the resampling filter can be computed efficiently

The Surface Splatting Algorithm

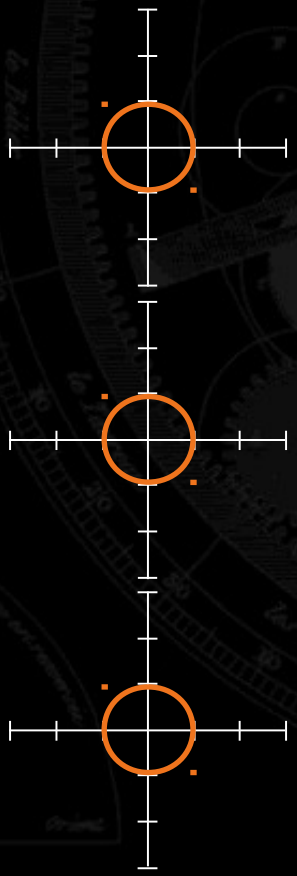
```
for each point P {
  project P to screen space;
  shade P;
  determine resampling kernel G;
  splat G[k];
}
```

Reconstruction Kernel Only

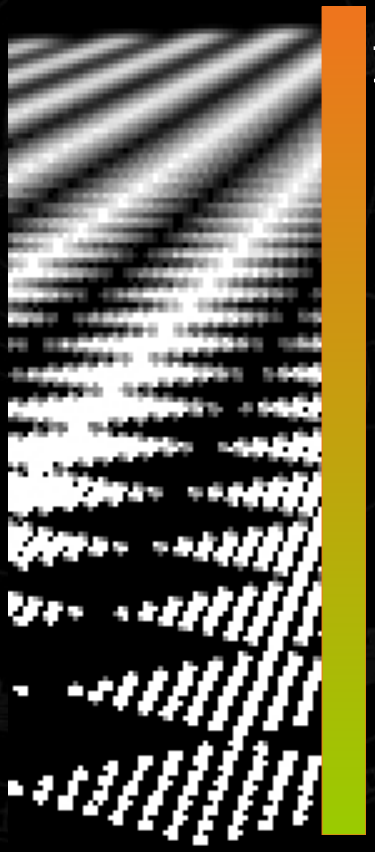


Low-Pass Filter Only

low-pass
filter



minification



no aliasing

holes

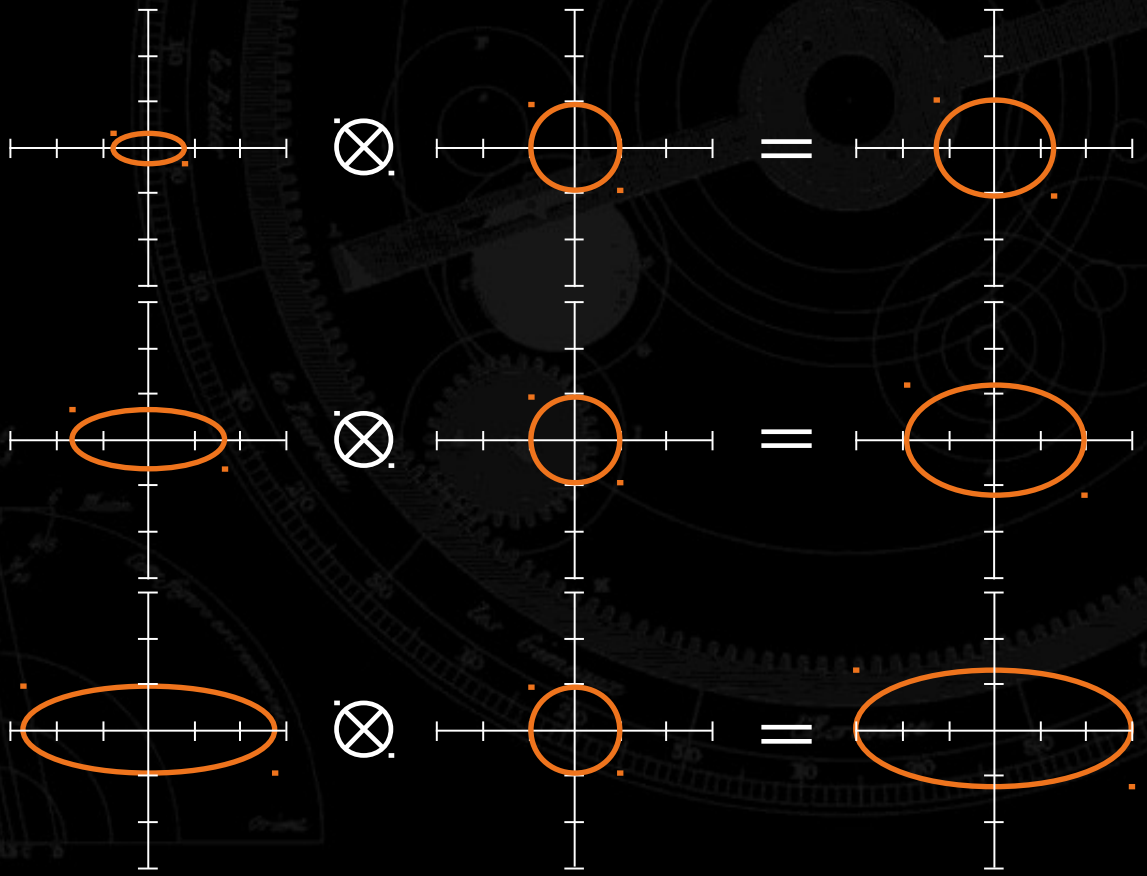
magnification

Screen Space EWA Properties

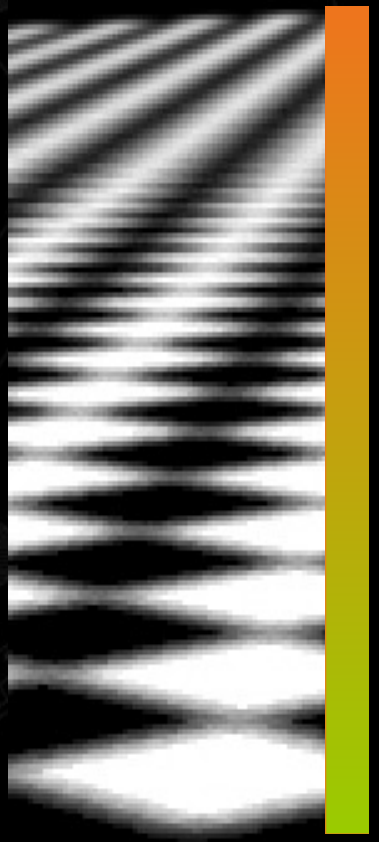
warped recon-struction kernel

low-pass filter

resampling filter

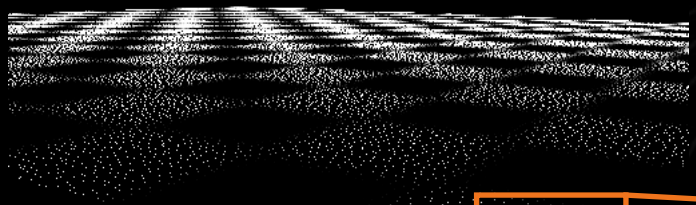


minification



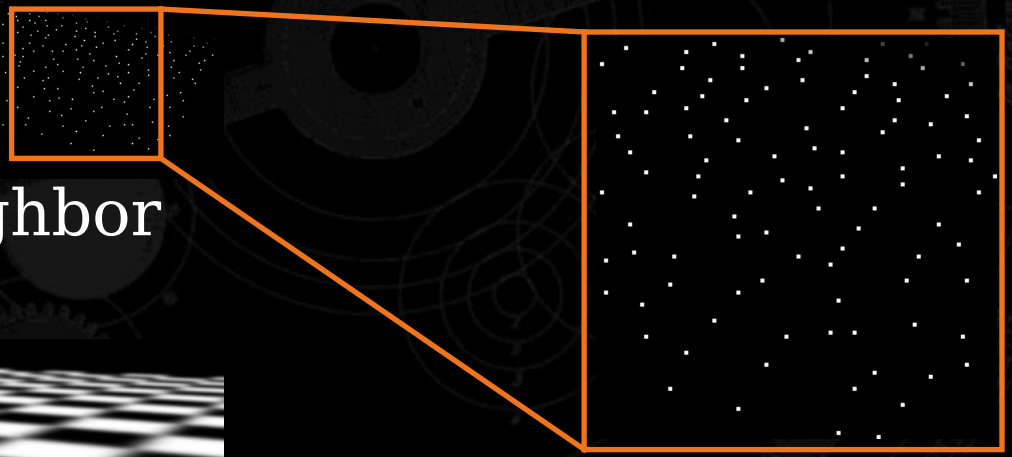
magnification

Irregular Textures



nearest neighbor

screen space EWA

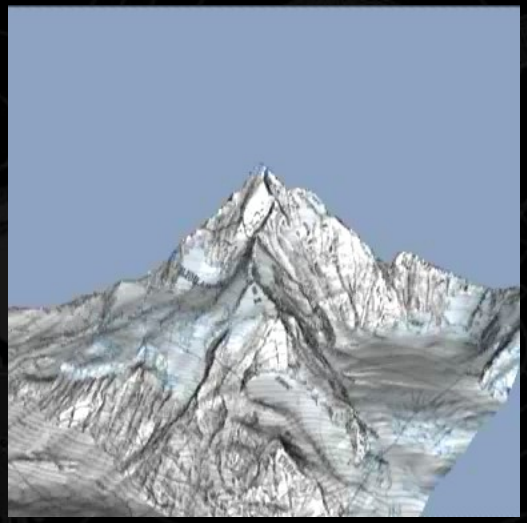


sampling pattern

Filter Normalization

- In the irregular setting, the resampling kernels do not sum up to one
- Solution alternatives:
  - In a pre-process, optimize the weights such that normalization is not necessary
  - Perform per pixel normalization after sampling at the pixel centers

Textured Digital Terrain



Further Issues

- Mathematical formulation of the resampling filter
- Details of the surface splatting algorithm
- Texture acquisition, weight computation

- Rendering semi-transparent surfaces
- Edge antialiasing

Conclusion: Surface Splatting

- Point rendering method with high-quality image reconstruction
- Based on Paul Heckbert’s EWA texture filter
- Anisotropic texture filtering for irregular point-sampled objects
- Transparency, edge antialiasing
- Can replace heuristics of previous splatting methods and provides superior texture quality

Future Work

- Computation of Gaussian reconstruction kernels
- Scanned Objects
- Compression
- Volume Rendering (IEEE Visualization 2001)
- Hardware acceleration

Acknowledgements

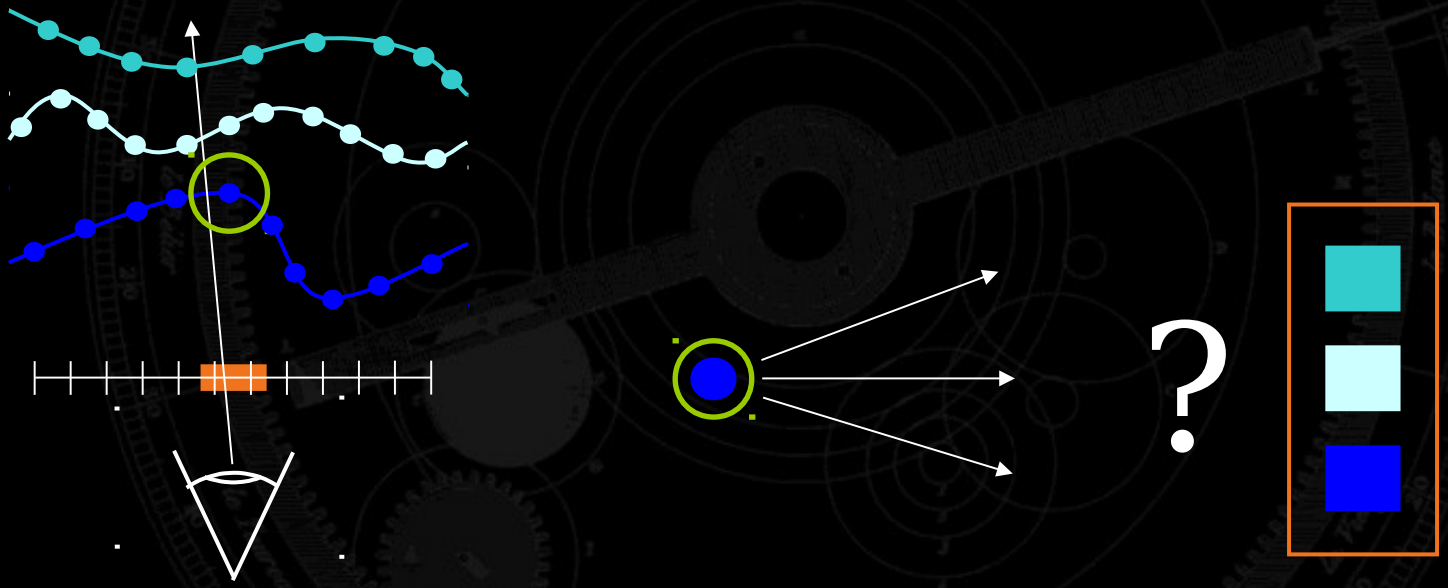
Paul Heckbert
Mark Pauly
Martin Roth
Jennifer Roderick
Marc Levoy and the
Digital Michelangelo Project

Matterhorn data set courtesy of *Bundesamt für Landestopographie,* Bern, Switzerland

Transparency

- Use modified A-buffer algorithm
- Contributions of each surface are accumulated in a separate bucket
- Challenge is to correctly decide to which bucket a new contribution belongs

Transparency



viewing ray new contribution pixel

- Extrapolate depth on tangent plane
- Use depth comparisons to find correct bucket
- Blend buckets back-to-front

Image Reconstruction

- Generate a raster image from projected points
- Avoid sampling artifacts (holes, aliasing)

holes

128 x 192

aliasing

128 x 192

no artifacts

128 x 192

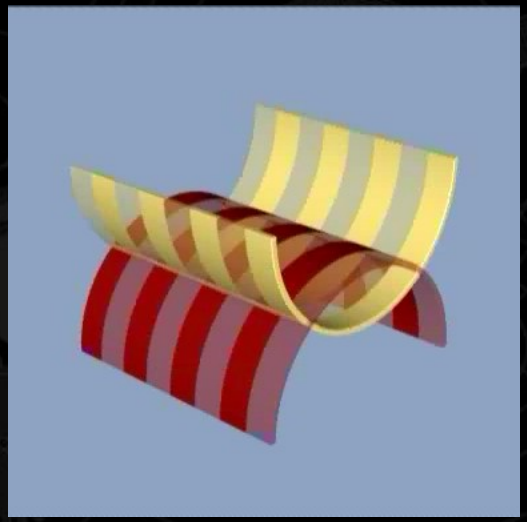
Edge Antialiasing

- The Gaussian resampling kernels approximate a partition of unity

$$\sum_k G_k(x) \approx 1$$

- Use the accumulated sum as a coverage coefficient
- Perform alpha blending using the coverage coefficients

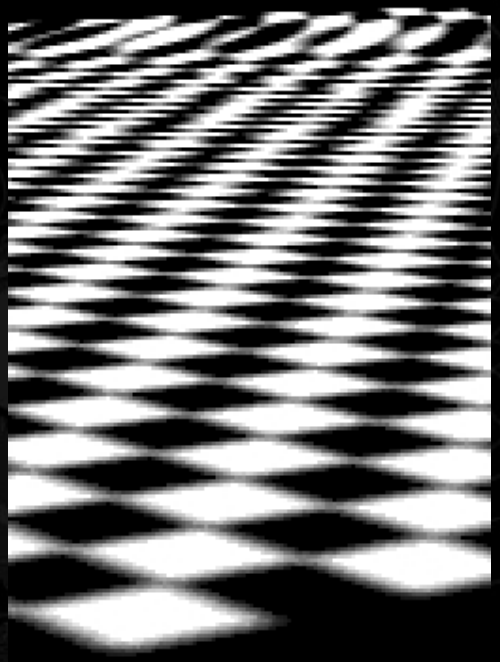
Semi-transparent Surfaces Edge Antialiasing



Elliptical Splats

minification

magnification



128 x 192

aliasing

smooth
reconstruction